Fast Re-Routing in Networks: On the Complexity of Perfect Resilience

Matthias Bentert (TU Berlin)
Esra Ceylan (TU Berlin + ISTA)
Valentin Huebner (ISTA)
Stefan Schmid (TU Berlin)
Jiri Srba (Aalborg University)

Acknowledgements:

W|W|T|F

erc
European Research Council
Established by the European Commission

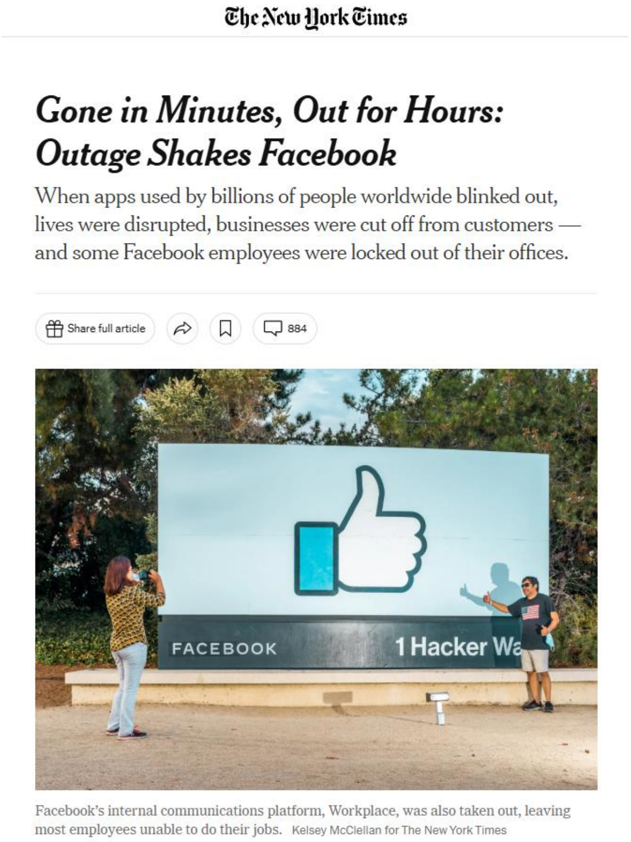
Networks:

Critical Infrastructure

- If networks break, it can have knock-on effects

- For example, Facebook outage in 2021: not only took down their social networking site, but also Instagram, WhatsApp, …

- … and their own internal systems, which manage the doors: engineers had to break into their own buildings to bring the network back up



The New York Times

Gone in Minutes, Out for Hours: Outage Shakes Facebook

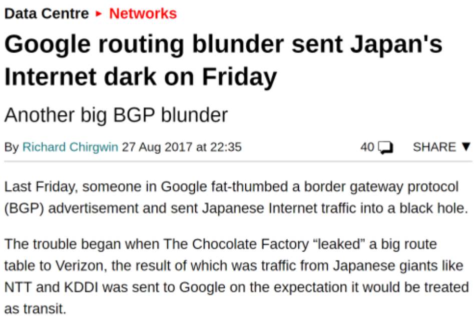
When apps used by billions of people worldwide blinked out, lives were disrupted, businesses were cut off from customers — and some Facebook employees were locked out of their offices.

Facebook's internal communications platform, Workplace, was also taken out, leaving most employees unable to do their jobs. Kelsey McClellan for The New York Times

Credits: Nate Foster

Human Errors

Countries disconnected



Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

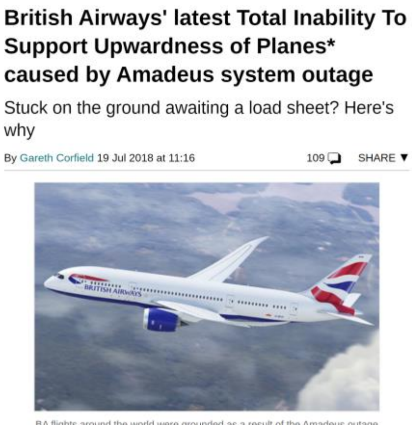
Another big BGP blunder

By Richard Chirgwin 27 Aug 2017 at 22:35 40 SHARE ▼

Last Friday, someone in Google fat-thumbed a border gateway protocol (BGP) advertisement and sent Japanese Internet traffic into a black hole.

The trouble began when The Chocolate Factory "leaked" a big route table to Verizon, the result of which was traffic from Japanese giants like NTT and KDDI was sent to Google on the expectation it would be treated as transit.

Passengers stranded



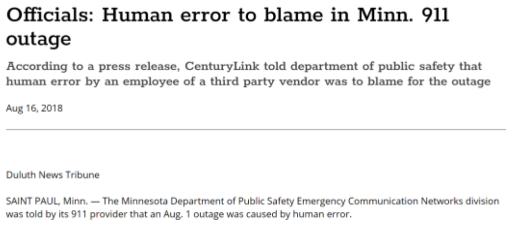
British Airways' latest Total Inability To Support Upwardness of Planes* caused by Amadeus system outage

Stuck on the ground awaiting a load sheet? Here's why

By Gareth Corfield 19 Jul 2018 at 11:16 109 SHARE ▼

BA flights around the world were grounded as a result of the Amadeus outage

Even 911 affected



Officials: Human error to blame in Minn. 911 outage

According to a press release, CenturyLink told department of public safety that human error by an employee of a third party vendor was to blame for the outage

Aug 16, 2018

Duluth News Tribune

SAINT PAUL, Minn. — The Minnesota Department of Public Safety Emergency Communication Networks division was told by its 911 provider that an Aug. 1 outage was caused by human error.

Even tech-savvy companies struggle:

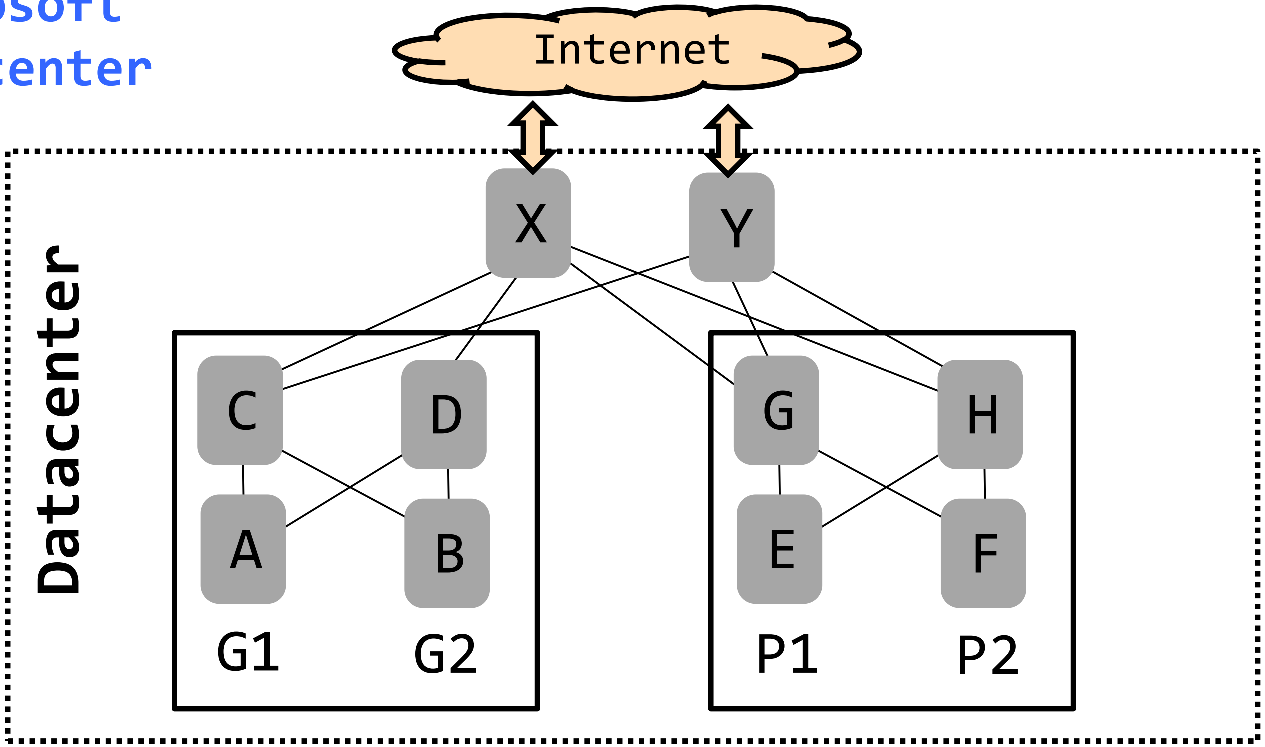
Mainly: human errors!

Slide credits: Nate Foster and Laurent Vanbever

A Reason: Complexity

Especially Under Failures (Policy Compliance)

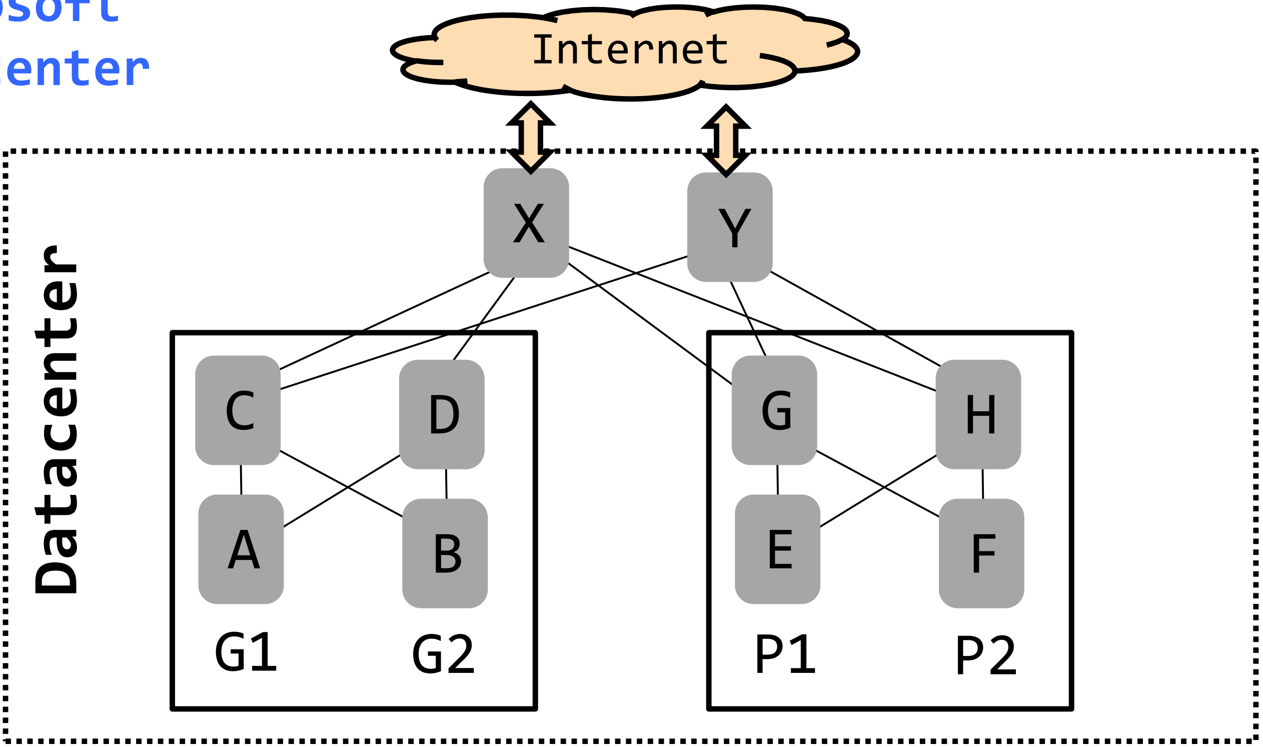
Example: BGP in **Microsoft datacenter**



A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



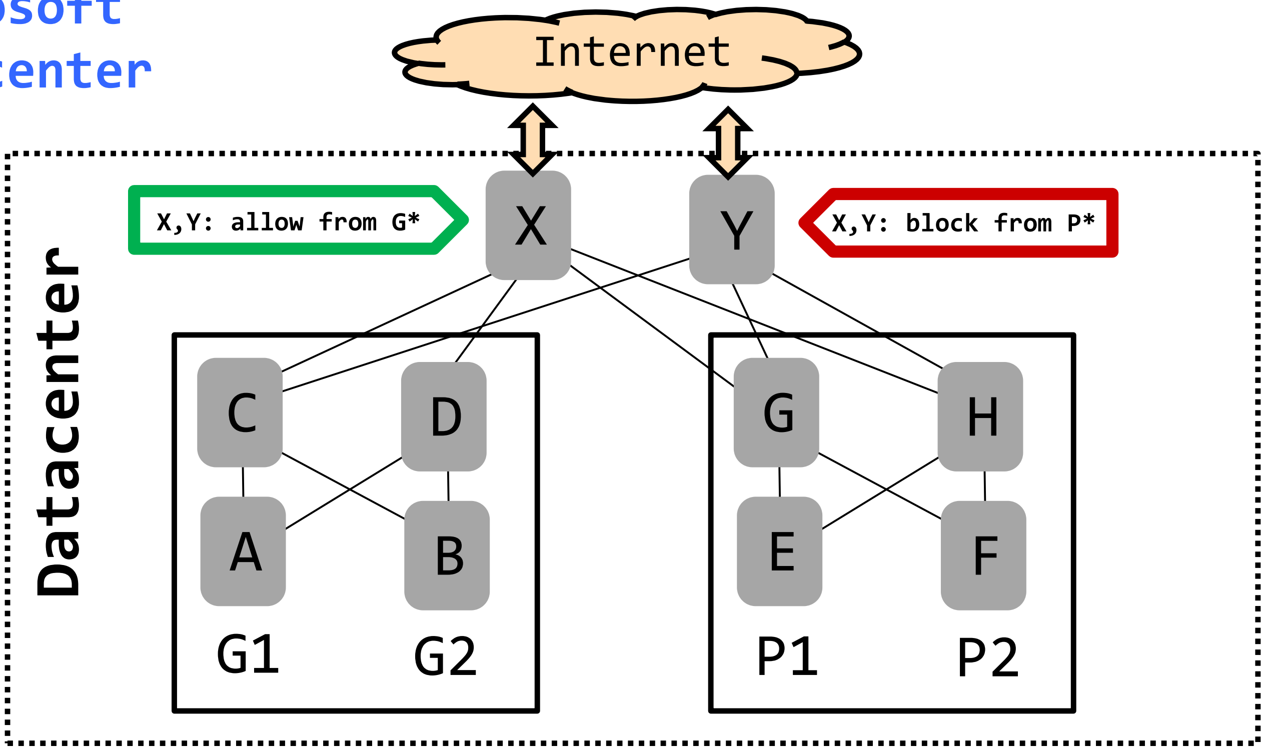
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



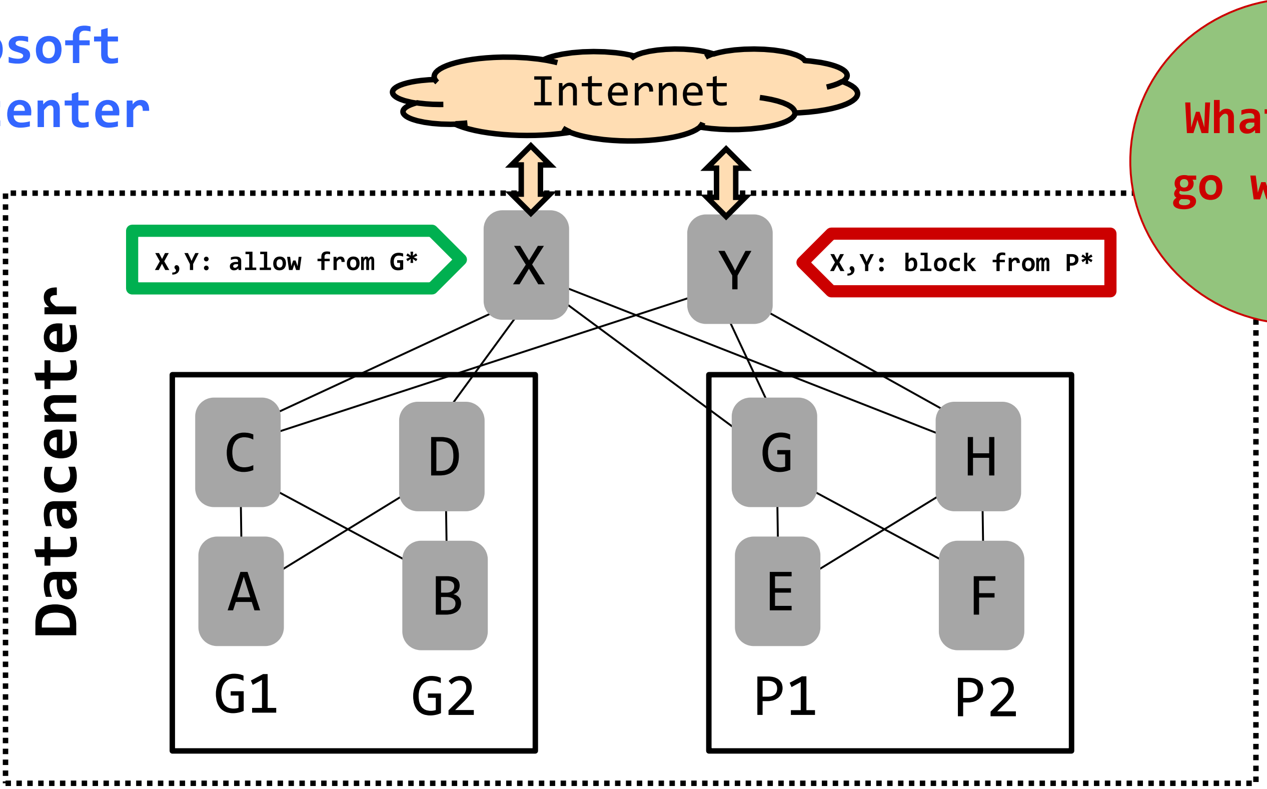
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



What can go wrong?

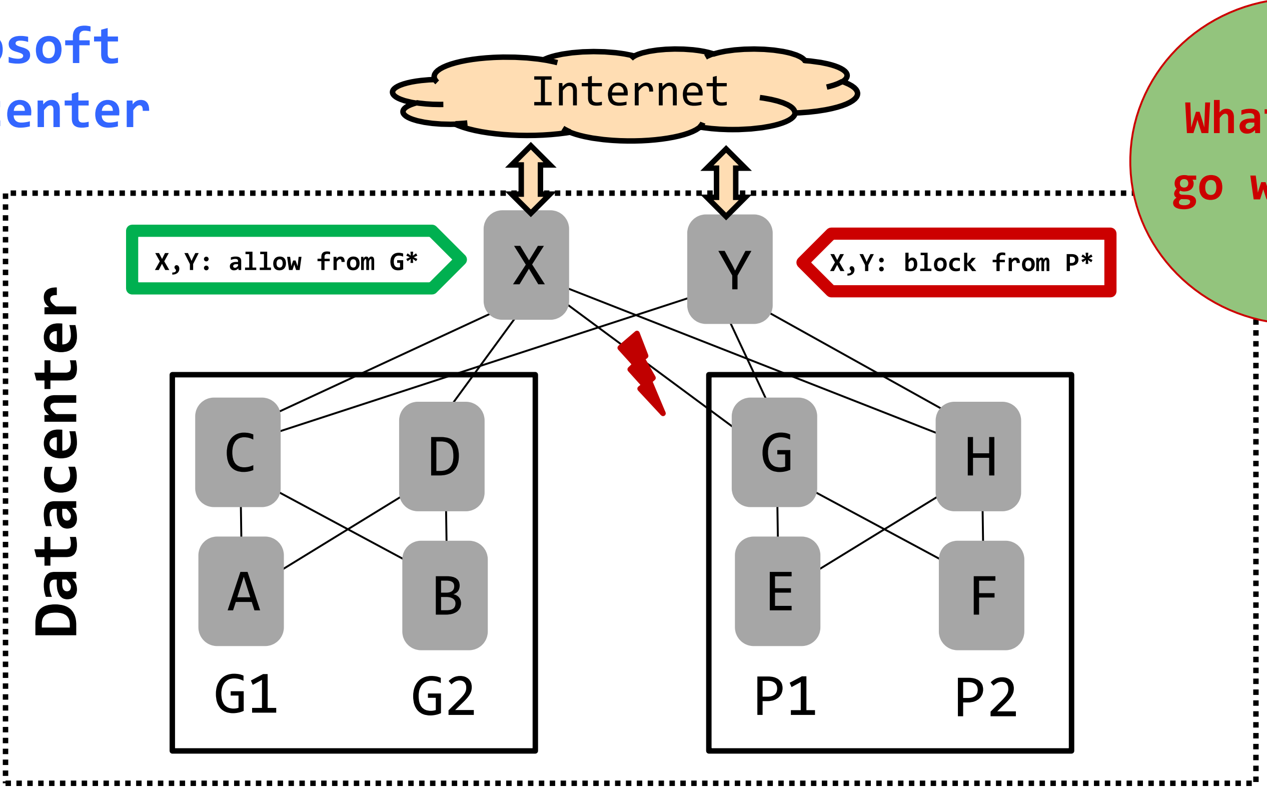
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**

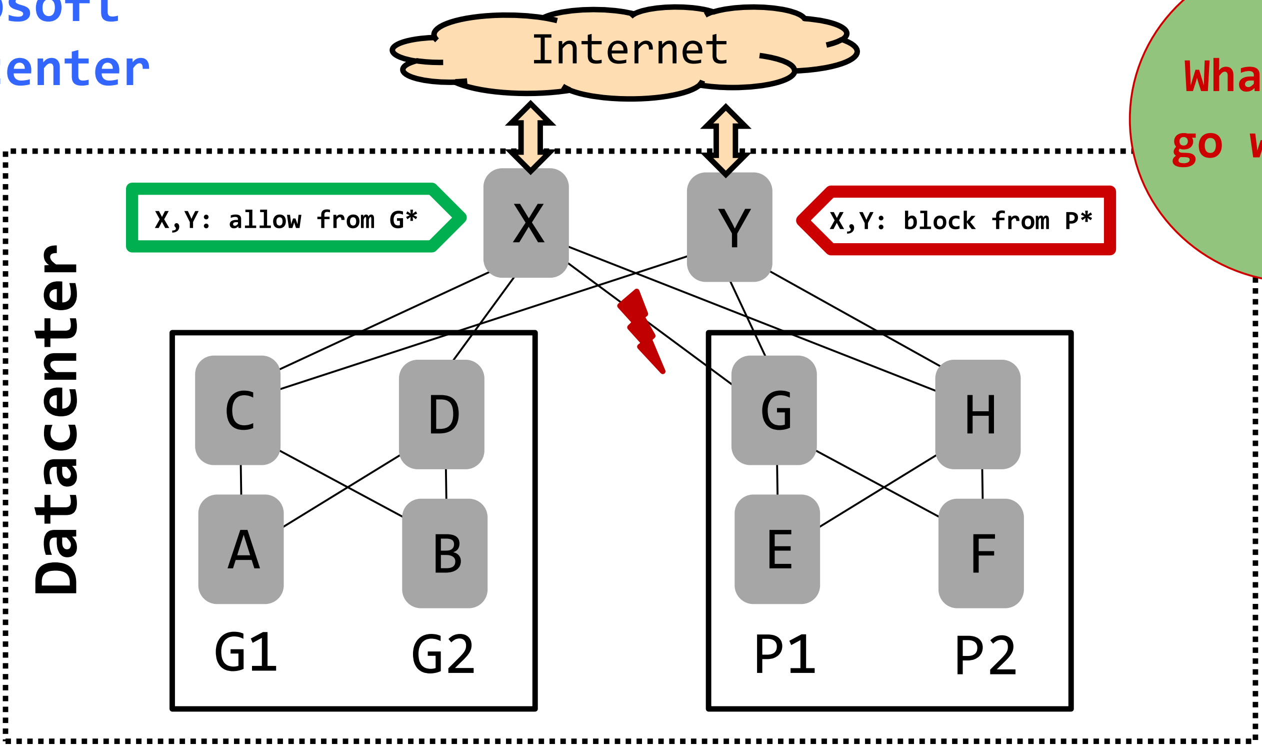


What can go wrong?

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in Microsoft datacenter



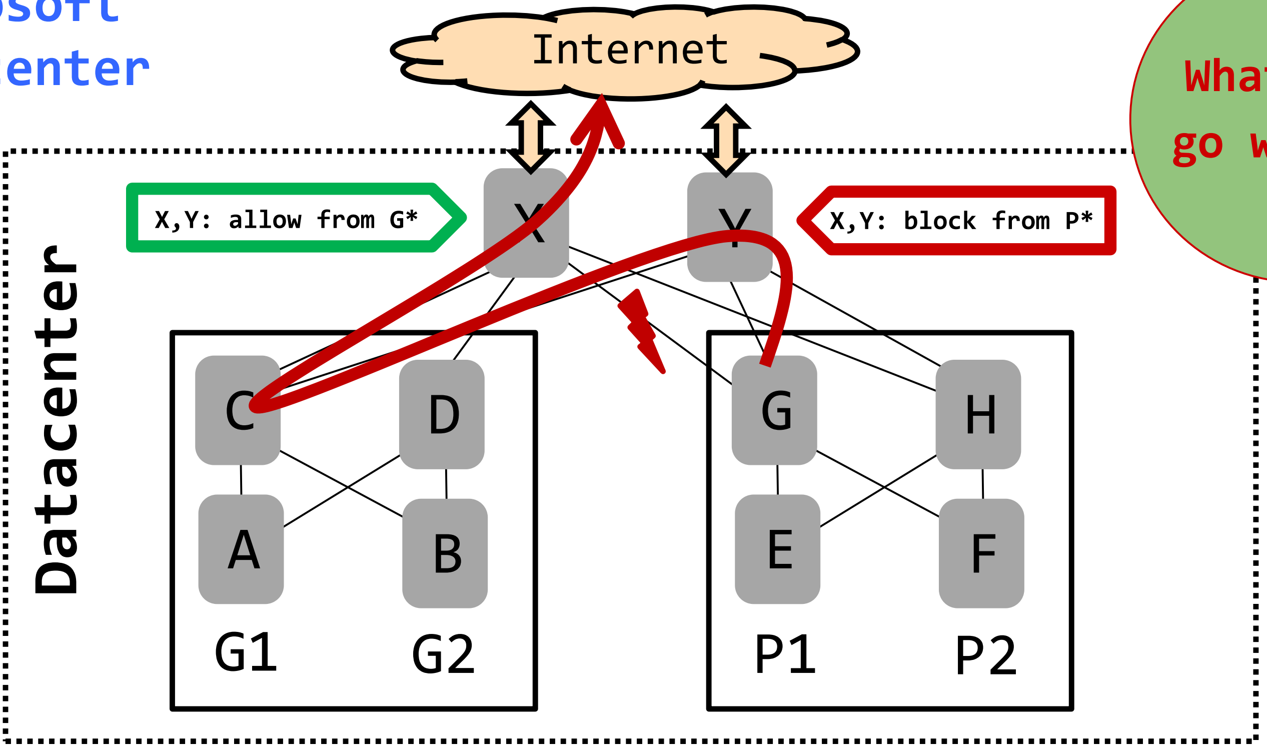
What can go wrong?

If link (G,X) fails and traffic from G is rerouted via Y and C to X:
X announces (does not block) G and H as it comes from C. (Note: BGP.)

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in Microsoft datacenter

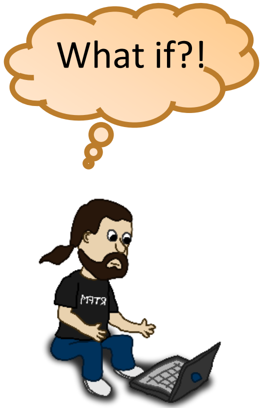


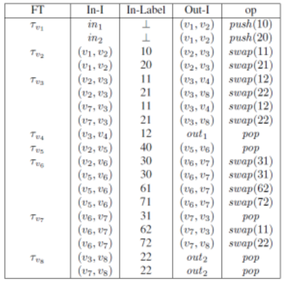
What can go wrong?

If link (G,X) fails and traffic from G is rerouted via Y and C to X:
X announces (does not block) G and H as it comes from C. (Note: BGP.)

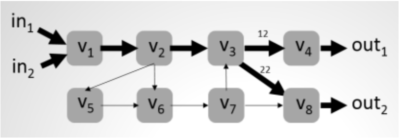
Vision to Deal with Complexity:

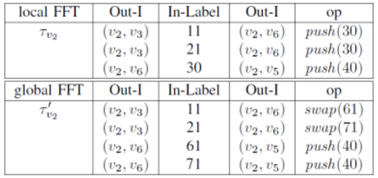
Automation





FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



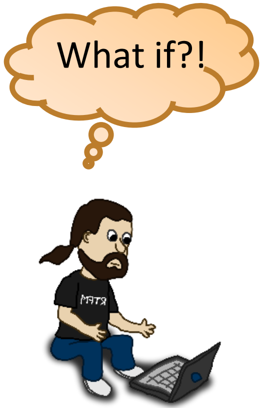


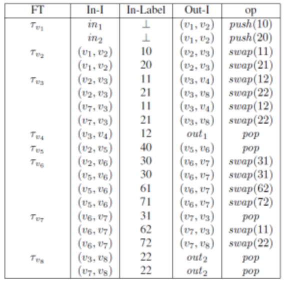
local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	**Out-I**	**In-Label**	**Out-I**	**op**
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Router configurations
(Cisco, Juniper, etc.)

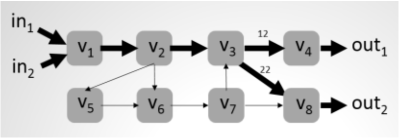
Vision to Deal with Complexity:

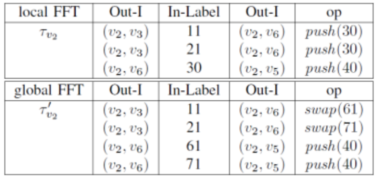
Automation





FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop





local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Compilation

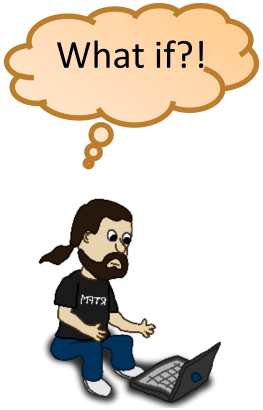
Resilient/
Compliant?

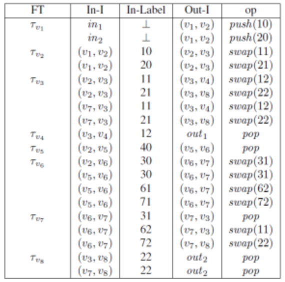
$$\mathrm{pX} \Rightarrow \mathrm{qXX}$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

Formal language which supports ***automated verification***
(„what-if analysis“)

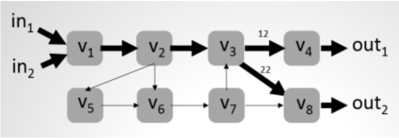
Vision to Deal with Complexity:

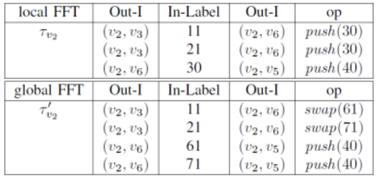
Automation





FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop





local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Compilation

$$pX \Rightarrow qXX$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

Or even *generate/fix*?

Formal language which supports ***automated verification***
(„what-if analysis“)

⇢ Generation aka. Synthesis

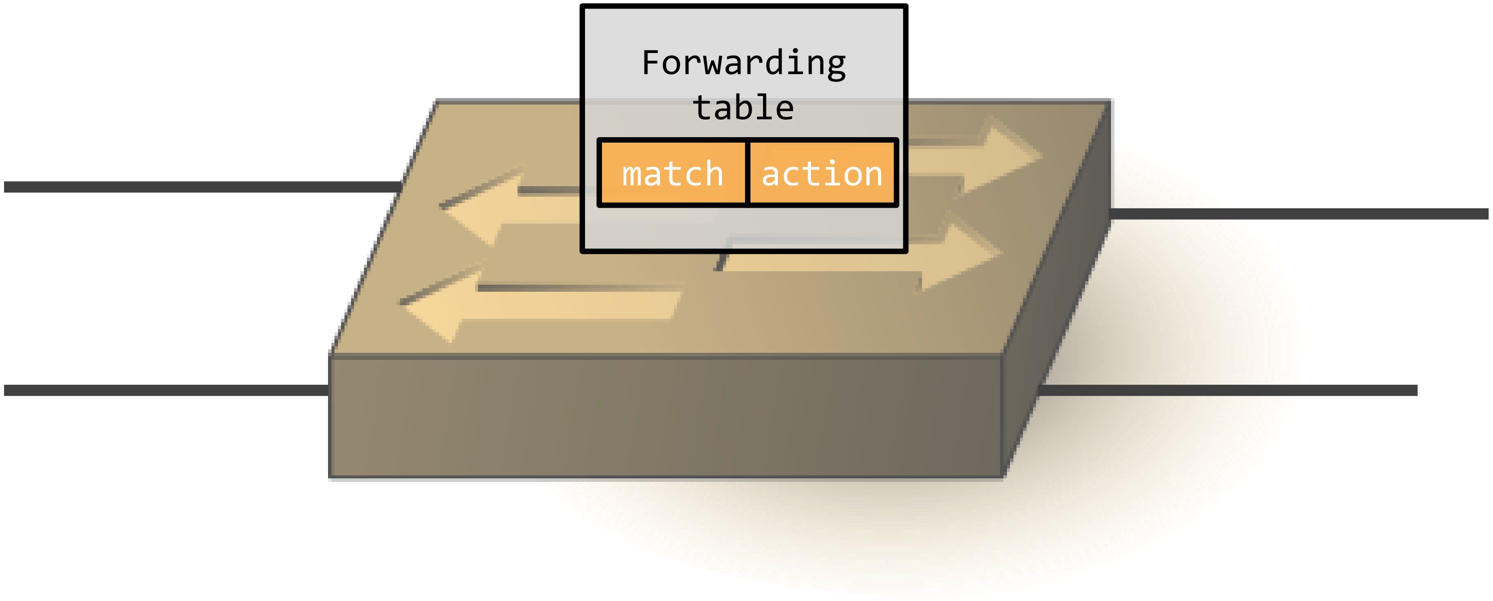
Information at Switch for

Local Decision Making?

Information at Switch for

Local Decision Making?

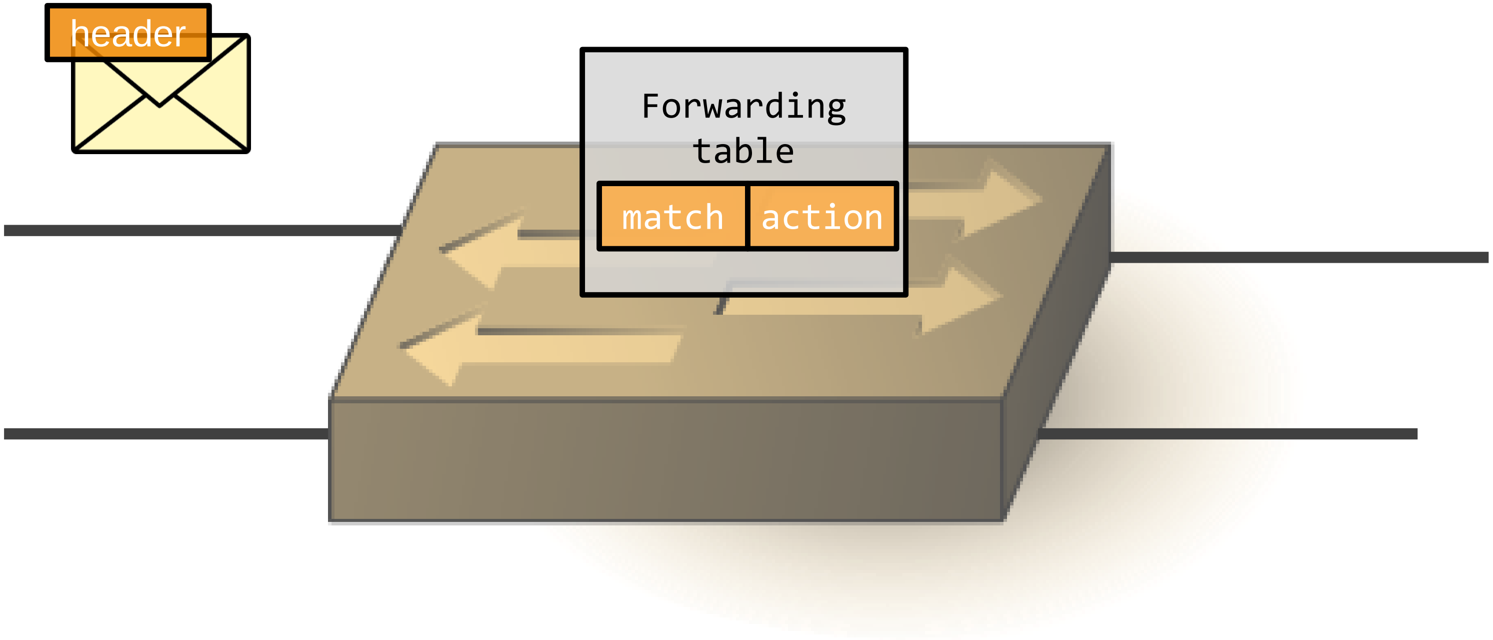
⇢ Nodes locally store a forwarding Match -> Action table



Information at Switch for

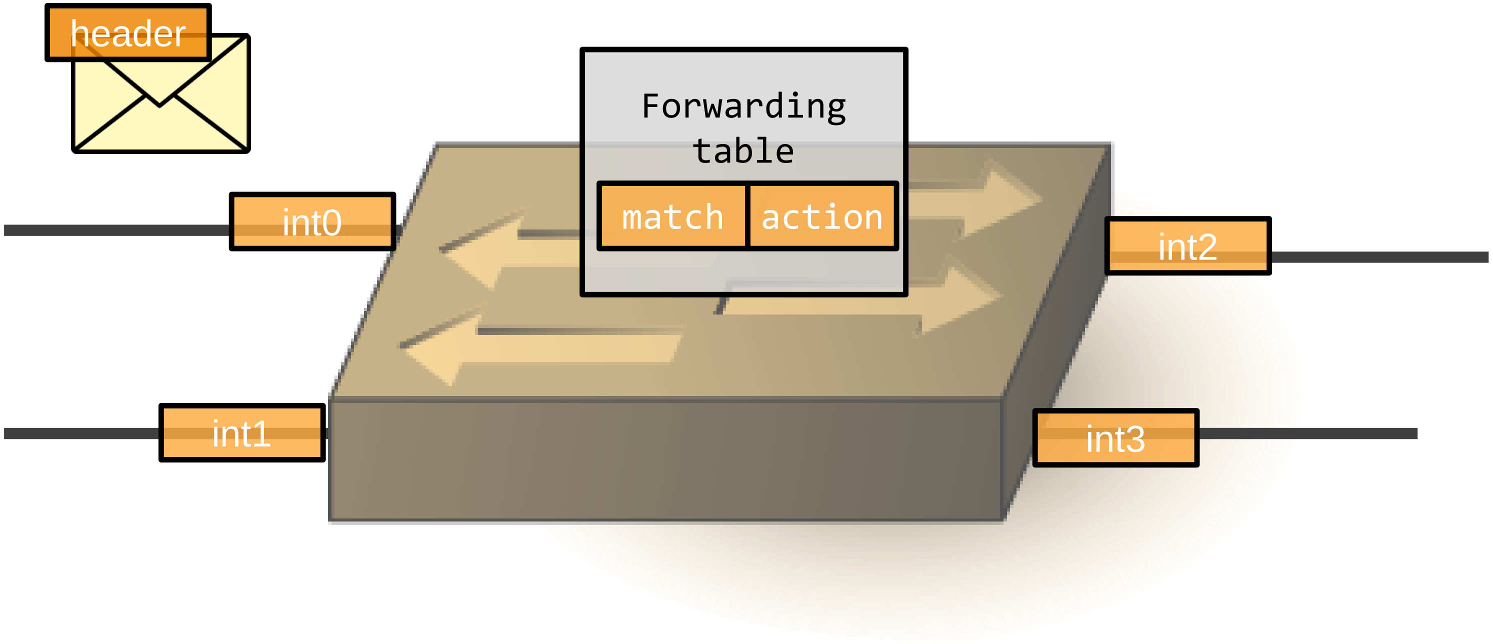
Local Decision Making?

⇢ The Packet Header (e.g., source, destination)



Information at Switch for
Local Decision Making?

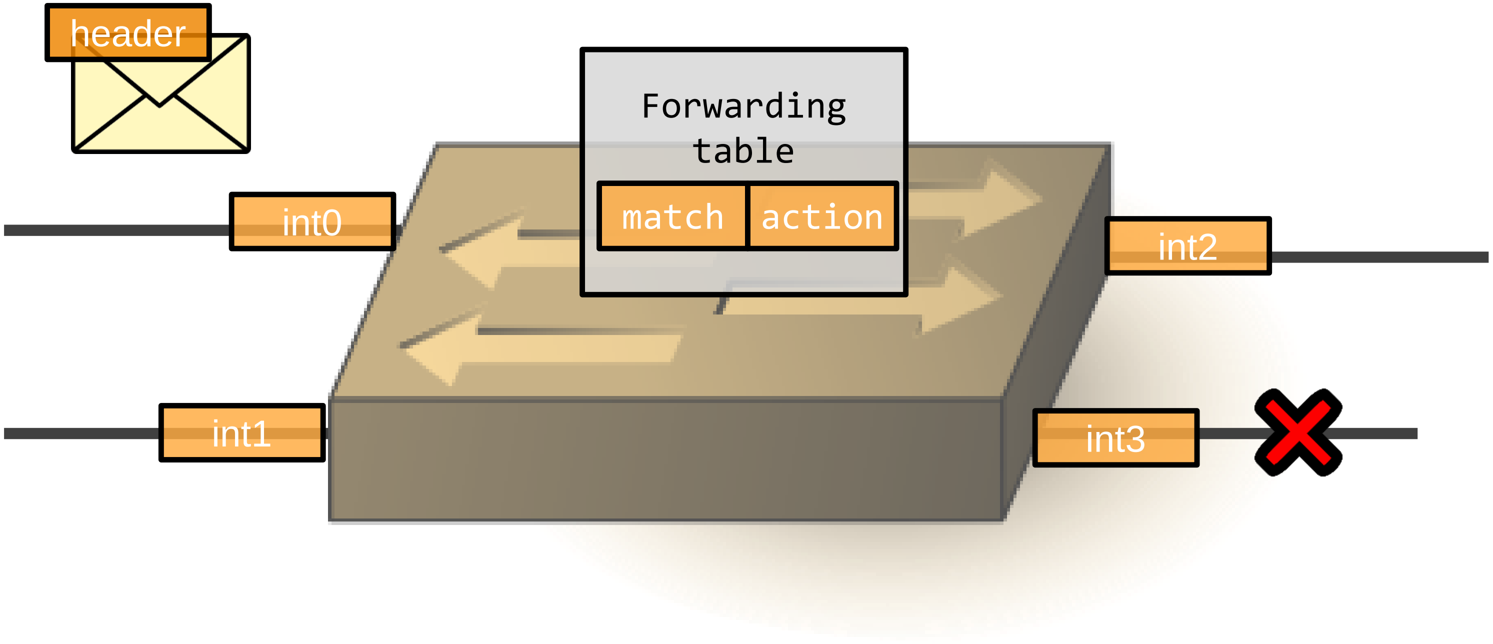
⇢ The Inport of the received packet



Information at Switch for

Local Decision Making?

⇢ Which incident links failed



Two fundamental

Notions of Resilience

Ideal resilience

Given a *k*-connected graphs, fast reroute can tolerate *any k-1 link failures.*

Perfect resilience

Fast reroute can tolerate any failures as long as the unterlying network is *physically connected*.

What is the difference? Which is stronger?

Two fundamental

Notions of Resilience

Ideal resilience

Given a *k*-connected graphs, fast reroute can tolerate *any k-1 link failures.*

Perfect resilience

Fast reroute can tolerate any failures as long as the unterlying network is *physically connected*.

- Our focus: *verification and synthesis* of *perfect resilience*
- We do have insights in verification of ideal resilience
- Synthesis of ideal resilience: *open problem* (harder)

Our Results

⇢ *Synthesis* of Perfect Resilience is ***in P***. (Non-constructive! Big open problem: *synthesis of perfect resilience*.)

⇢ *Verification* of Perfect Resilience is *coNP-complete*. Even if the input graph is *planar* and for simple "*skipping* rules".

⇢ *Verification and synthesis* problems for Perfect Resilience using *in-port oblivious* are in *linear time* (constructive).

Our Results

- *Synthesis* of Perfect Resilience is ***in P***. (Non-constructive! Big open problem: *synthesis of perfect resilience*.)

- *Verification* of Perfect Resilience is *coNP-complete*. Even if the input graph is *planar* and for simple “*skipping* rules”.

- *Verification and synthesis* problems for Perfect Resilience using *in-port oblivious* are in *linear time* (constructive).

There exists a perfectly resilient in-port oblivious local (or skipping) routing for any given target node if and only if all simple *cycles* of G have length *at most 3*. Can be computed with a modified BFS.

Our Result

From *Robertson and Seymour*’s result on the *minor-stability* property for the graphs for which perfect resilience is not impossible (known from *Foerster et al.*). Whether a graph contains a *forbidden minor* can be checked in polynomial time.

- *Synthesis* of Perfect Resilience is ***in P***. (Non-constructive! Big open problem: *synthesis of perfect resilience*.)

- *Verification* of Perfect Resilience is *coNP-complete*. Even if the input graph is *planar* and for simple “*skipping* rules”.

- *Verification and synthesis* problems for Perfect Resilience using *in-port oblivious* are in *linear time* (constructive).

Our Results

- *Synthesis* of Perfect Resilience is ***in P***. (Non-constructive! Big open problem: *synthesis of perfect resilience*.)

Next!

- *Verification* of Perfect Resilience is *coNP-complete*. Even if the input graph is *planar* and for simple “*skipping* rules”.

- *Verification and synthesis* problems for Perfect Resilience using *in-port oblivious* are in *linear time* (constructive).

CoNP Completeness

Recall *coNP-complete*: complement of the language is *NP-complete*
⇢ Alternative to show hardness: reduce from coNP-hard problem

The problem is in coNP:
⇢ We can *guess a failure* scenario where a source node is connected to the target but where the routing tables *create a forwarding loop*
⇢ The given failure scenario can be checked in polynomial time

coNP-hardness:
⇢ By a reduction from 3-Sat with exactly 3 literals
⇢ The constructed skipping routing is perfectly resilient if and only if the original instance of 3-Sat is *not satisfiable*

Future Work

⇢ Ideal resilience conjecture

⇢ Constructive proof that perfect resilience synthesis is in P

Thank you and References

A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks
Marco Chiesa, Andrzej Kamisinski, Jacek Rak, Gabor Retvari, and Stefan Schmid.
IEEE Communications Surveys and Tutorials (**COMST**), 2021.

AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks
Peter Gjøl Jensen, Morten Konggaard, Dan Kristiansen, Stefan Schmid, Bernhard Clemens Schrenk, and Jiri Srba.
16th ACM International Conference on emerging Networking EXperiments and Technologies (**CoNEXT**), Barcelona, Spain, December 2020.

A Tight Characterization of Fast Failover Routing: Resiliency to Two Link Failures is Possible
Wenkai Dai, Klaus-Tycho Foerster, and Stefan Schmid.
35th ACM Symposium on Parallelism in Algorithms and Architectures (**SPAA**), Orlando, Florida, USA, June 2023.

On the Price of Locality in Static Fast Rerouting
Klaus-Tycho Foerster, Juho Hirvonen, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan.
52nd IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), Baltimore, Maryland, USA, June 2022.

SyPer: Synthesis of Perfectly Resilient Local Fast Rerouting Rules for Highly Dependable Networks
Csaba Györgyi, Kim G. Larsen, Stefan Schmid, and Jiri Srba.
IEEE Conference on Computer Communications (**INFOCOM**), Vancouver, Canada, May 2024.

DeepMPLS: Fast Analysis of MPLS Configurations Using Deep Learning
Fabien Geyer and Stefan Schmid.
IFIP Networking, Warsaw, Poland, May 2019.

Latte: Improving the Latency of Transiently Consistent Network Update Schedules
Mark Glavind, Niels Christensen, Jiri Srba, and Stefan Schmid.
38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (**PERFORMANCE**) and ACM Performance Evaluation Review (**PER**), Milan, Italy, November 2020.

Model-Based Insights on the Performance, Fairness, and Stability of BBR
Simon Scherrer, Markus Legner, Adrian Perrig, and Stefan Schmid.
ACM Internet Measurement Conference (**IMC**), Nice, France, October 2022.

Slides available: